"eL TIMoN": una herramienta basada en morfismos para la enseñanza de la programación orientada a objetos

Inés Friss de Kereki, Carlos Nicolás Fornaro
Facultad de Ingeniería, Universidad ORT Uruguay
Montevideo, 11100, Uruguay
{kereki_i, fornaro}@ort.edu.uy

y

Javier Azpiazu, José Crespo
Facultad de Informática, Universidad Politécnica de Madrid,
Madrid, 28800, España
{jazpiazu, jcrespo}@fi.upm.es

Abstract

Software design implies establishing a correspondence between real world and a software model. Morphisms establish correspondences between different domains. It is considered that use of morphisms can be useful for development of mental models and learning strategies to analize and build adequate software models. The first part of this investigation was done in Universidad ORT Uruguay, during 2005. There, it was verified that the use of morphisms helps to develop said models and strategies. The follow up of this investigation is presented in this paper. Components and elements needed for integrating information and communication technologies and use of morphisms in a "e-learning" model were defined. A specific tool, named "eL TIMóN" (e-Learning Tool for Instruction based on Morphisms' Notions, "el timón" meaning "the helm" in Spanish) was implemented. Use of this tool has been proved useful for developing and building of adequate software models.

Keywords: teaching, learning, e-learning, object oriented programming, first year course.

Resumen

El diseño de software implica establecer una correspondencia entre el mundo real y el modelo de software. Los morfismos establecen correspondencias entre diferentes dominios. Se plantea que el uso de los morfismos puede colaborar en el desarrollo de modelos mentales y estrategias de aprendizaje para analizar y construir modelos de software adecuados. En 2005 en Universidad ORT Uruguay se realizó una primera parte de esta investigación. De ella se constató que el uso de morfismos colabora en el desarrollo de dichos modelos y estrategias. En este trabajo se presenta la continuación de la investigación. Se definieron los componentes y elementos necesarios en un modelo de "e-learning" que integre las tecnologías de la información y comunicaciones y que incorpore el uso de morfismos. Se desarrolló una herramienta específica denominada "eL TIMoN" (e-Learning Tool for Instruction based on Morphisms' Notions). A partir de los resultados de la experimentación, se infiere que el uso de la herramienta resulta beneficioso para el desarrollo y construcción de modelos adecuados de software.

Palabras claves: enseñanza, aprendizaje, *e-learning*, programación orientada a objetos, primera asignatura.

1 INTRODUCCION

El aprendizaje es un proceso constructivo que se produce cuando lo que se le enseña es útil y significativo para el aprendiz y cuando éste participa activamente en la adquisición de sus propios conocimientos, relacionando lo que está aprendiendo con conocimientos y experiencias anteriores. Los alumnos aprenden mejor cuando son parte de la ecuación del aprendizaje [16].

La educación, en vez de centralizarse en cursos y calificaciones, se debe enfocar en los fines y metas, como por ejemplo: autonomía, responsabilidad y creatividad. Por creatividad refiere a la capacidad de pensar y actuar creativamente, resolver problemas, entender estructuras y disciplinas y aplicar el conocimiento en forma que se extienda y desarrolle [4].

Aprender a aprender refiere a lo que debe hacerse para equipar al alumno con el instrumental del autoaprendizaje. La meta es enseñar al individuo a razonar, a emitir juicios, a elaborar argumentos sólidos, a criticar los argumentos de otros, a familiarizarse con el uso de todos los tipos de materiales didácticos (libros, bibliotecas, computadoras, etc). En resumen, enseñar al alumno a pensar y actuar en forma autónoma a fin de que adquiera más conocimientos con creciente capacidad y destreza [13]. Dentro de la batería de herramientas necesarias o útiles para estas tareas podrían incluirse las ideas poderosas.

Se entiende por idea poderosa toda idea, concepto o elemento con pretensión de perdurar en el tiempo, de largo alcance, dominio y amplio rango [19]. Pazos [18] citando a Papert refiere al concepto de ideas poderosas, entre las que se destacan los morfismos, las ducciones (deducciones, inducciones y abducciones) y la recursión. En particular, los morfismos establecen correspondencias entre distintos dominios mientras se mantienen las propiedades. Usar morfismos permite ver diferentes cosas como la misma y tomar la imagen sustituta por la real.

En este trabajo se presenta la utilización explícita de morfismos como una herramienta útil para el desarrollo de modelos. En particular se aplicó el uso de morfismos a la enseñanza inicial de la programación orientada a objetos.

2 ENSEÑANZA DE LA PROGRAMACIÓN

Programar es una actividad intelectual compleja [22]. Robins y colegas indican que aprender a programar implica adquirir conocimientos nuevos completos y las estrategias relacionadas, así como habilidades prácticas [20]. Las deficiencias más importantes que presentan los novicios están vinculadas a las actividades de resolver problemas, diseñar y expresar el diseño o la solución como un programa.

Mc. Cracken y colegas [17] señalan que los objetivos de un curso de 1er. año de carreras de computación incluyen el proceso de resolver problemas en el dominio de la ciencia de la computación, con la finalidad de producir programas ejecutables que sean correctos y en una forma apropiada. Se trata de abstraer, generar subproblemas, transformarlos en subsoluciones, recomponer integrando esas subsoluciones y finalmente, evaluar e iterar este proceso hasta llegar a una solución buena. Indican que los estudiantes a menudo perciben que el foco en este tipo de curso es aprender la sintaxis y no se concentran en planificar o diseñar.

Kalling y colegas recomiendan presentar análisis y diseño orientado a objetos desde el comienzo y que los problemas deben ser analizados y entendidos antes de codificar una solución [14]. También señalan que enseñar resolución de problemas orientado a objetos ha resultado ser más difícil de lo esperado. Indican como principales problemas en los cursos iniciales de programación orientada a objetos la falta de pedagogías probadas para enseñar esos conceptos y la falta de entornos adecuados de programación. Muchos entornos son complicados para los que recién comienzan.

Según indican Truong y colegas [22] los programas construidos por estudiantes de primer año a menudo están pobremente elaborados pues no consideran diferentes soluciones. Para muchos estudiantes entender los conceptos como clases y objetos es difícil [10].

Hay muchos enfoques para enseñar programación. Algunos enfatizan las clases teóricas frente a las prácticas, el trabajo individual frente al colaborativo [11], otros se basan en herramientas de simulación [10], uso de entornos especiales ([2], [8] [9], [21]), uso de juegos [3], programación por pares [1], juegos de rol [5], y, o, tarjetas CRC [6]. Bruce [7] refiere a que actualmente hay información insuficiente para evaluar cuán efectivas son las herramientas pedagógicas tales como entornos pedagógicos, bibliotecas especiales que proveen clases útiles o micromundos en la enseñanza de los cursos iniciales.

En particular, Jeroo [21] es una herramienta diseñada para aprender las nociones básicas de objetos, para resolver problemas y aprender a escribir métodos basado en la descomposición funcional de la tarea. Alice [9] se centra en la visualización de los objetos y sus comportamientos en un entorno animado de 3 dimensiones. En ambas herramientas, la dificultad es que los objetos a utilizar ya están definidos, no se buscan nuevas clases ni relaciones, tarea fundamental al diseñar software orientado a objetos.

Otro enfoque se centra en que, para ser programador, se deben desarrollar habilidades de escribir, leer, depurar y probar código. Bailey [2] propone que aprender a escribir programas pequeños y correctos desde el comienzo fortalece esas habilidades. RAPTOR [8] es un entorno visual diseñado para ayudar al desarrollo de algoritmos y evitar la carga de sintaxis. Se promueve mejorar las habilidades de resolución de problemas y disminuir la carga sintáctica inherente a la mayoría de los lenguajes de programación. Estos estudios se centran más bien en la programación en detalle pero no en el previo análisis y diseño de clases necesario.

El uso de juegos, como refieren Bayliss y colegas [3], también puede ser considerado como alternativa, integrándolos tanto al material teórico como al práctico. Sugieren que no todo el curso debe ser basado en juegos, pues hay personas que no les agrada.

La programación por pares [1] es otra herramienta más a utilizar, permite mejorar el entendimiento de los estudiantes pero la calidad del trabajo final fue menor que la esperada, según indican Ahern y colegas.

En resumen, del relevamiento bibliográfico se constata que hay diversas dificultades en la enseñanza inicial de la programación, algunas relacionadas con la pedagogía, otras con la comprensión de los conceptos y su representación, así como con las capacidades y estrategias de resolución de problemas. Como señalan Huet y colegas [11], siempre se trata de buscar mejorar la experiencia del aprendizaje a través de la reflexión y probando nuevas ideas para favorecer el éxito del aprendiz.

En ese marco, se propone el uso de morfismos como una herramienta que puede colaborar en el desarrollo de modelos mentales y en los procesos para analizar y diseñar software. Desde 2005 se realiza en Universidad ORT Uruguay una investigación acerca del uso de morfismos en la enseñanza de la programación orientada a objetos. El morfismo establece una correspondencia entre diferentes dominios, mientras se mantienen las propiedades. Un modelo es isomorfo respecto al sistema que trata de reproducir cuando el dominio original y el modelo pueden ser considerados equivalentes. En un sentido amplio y con enfoque informático, en vez de considerar que en los morfismos se conservan las operaciones, se podrían considerar que conservan los comportamientos [15].

En esa investigación [15], se hizo una comparación empírica de dos metodologías de aprendizaje diferentes: una basada en el aprendizaje de contenidos teóricos y prácticos impartidos con un enfoque fundamentalmente descriptivo, frente a otra alternativa equivalente a la anterior en cuanto a su contenido teórico, pero incorporando ideas derivadas de los morfismos como instrumento para mejorar el rendimiento del aprendizaje de los estudiantes. Los resultados experimentales mostraron que la capacitación en morfismos influye positivamente en el desarrollo de habilidades para modelado de problemas .

En 2006 se continuó la investigación con el objetivo de desarrollar un sistema de software que se base en el uso de los morfismos. En este trabajo se presenta la definición de los componentes y elementos necesarios en un modelo de "e-learning" que integre las tecnologías de la información y comunicaciones y que incorpore el uso de morfismos. Se diseñó una arquitectura para un modelo de "e-learning" basado en morfismos y se desarrolló una herramienta específica denomida "eL TIMoN" (*e-Learning Tool for Instruction based on Morphisms' Notions*). En este trabajo se presenta la arquitectura, el uso de la herramienta y la experimentación realizada.

4 PROGRAMACIÓN I EN UNIVERSIDAD ORT URUGUAY

El curso de Programación I consta de 15 semanas, en las cuales se dictan 4 horas de clase teórica y 2 horas de práctica en laboratorio por semana. El objetivo de la materia es iniciar la enseñanza de la programación empleando fundamentalmente técnicas de programación orientada a objetos. Se da énfasis a la enseñanza de una metodología de resolución de problemas. Capacita al estudiante para desarrollar aplicaciones sencillas con lenguajes orientados a objetos. Al final del curso el estudiante estará capacitado para analizar situaciones simples, diseñar una posible solución e implementarla basándose en un enfoque orientado a objetos. Como lenguaje de programación se utiliza Java desde 1999. El plan detallado por semanas se presenta en la Tabla 1.

Para aprobar el curso se deben realizar aceptablemente 2 trabajos obligatorios y un parcial. Cada trabajo obligatorio se hace en equipo de 2 estudiantes y dura un mes. El primer obligatorio implica generalmente el desarrollo de un pequeño sistema (3-4 clases) y el segundo es una ampliación y modificación del primero. En este caso, se agregan otras clases y procesos varios, con algoritmia interesante. Por cada trabajo hay una defensa individual. Además cada estudiante debe superar una prueba parcial escrita. Los trabajos obligatorios valen respectivamente 25 y 30 puntos. El parcial vale 45 puntos. Para la aprobación se requiere 70 puntos. Si se obtienen 86 o más puntos, se exonera la material, por lo cual no es necesario rendir examen final.

Tabla 1 Programación I

Semana	Temas
1-3	Variables, estructuras de control, seudocódigo
4	Presentación de clases y objetos, uso de clases standard
5-8	Creación de clases, alias, relaciones
9	Herencia, mutuo conocimiento de clases y objetos
10-12	Colecciones, Excepciones, Ordenación y búsqueda
13	Enumeración
14-15	Manejo avanzado de colecciones

5 "eL TIMoN"

5.1 Arquitectura de "eL TIMoN"

Como base para la arquitectura de "eL TIMoN" se tomó el estándar IEEE P1484.1-2003 [12]. Este estándar especifica una arquitectura de alto nivel para el aprendizaje, la educación y los sistemas de entrenamiento soportados a través de tecnologías de la información. En la figura 1 se presenta la arquitectura de la herramienta.

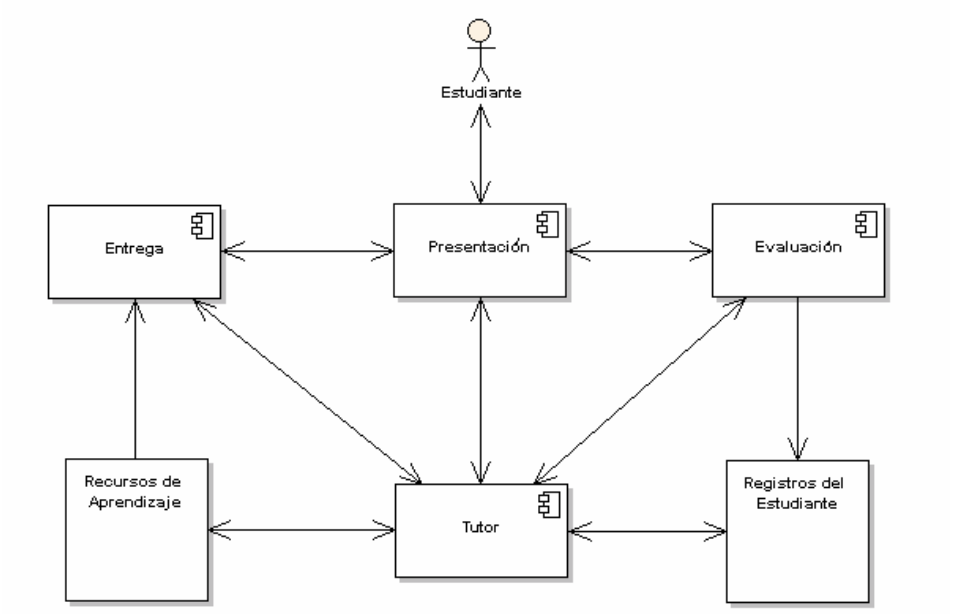

Figura 1: Arquitectura de "eL TIMoN"

Los componentes que incluye "eL TIMoN" son:

a) Estudiante: Cada estudiante se representa en el sistema como una entidad individual e independiente de los demás. Los estudiantes se identifican al iniciar la sesión lo que permite asociar las acciones realizadas a un perfil de usuario y registrar los datos históricos de todas las acciones realizadas por cada alumno al trabajar.

b) Evaluación: Este componente implementa la lógica necesaria para evaluar las soluciones realizadas por el estudiante, compararlas con las preexistentes y determinar su grado de corrección y completitud. A medida que el estudiante avanza en la resolución de cada ejercicio y lo solicita, el sistema compara esta solución con cada una de las soluciones registradas por los docentes, asignando puntajes preestablecidos para cada acierto o falla en la solución del estudiante. Estos puntajes los puede modificar el docente.

El proceso de evaluación verifica si las clases identificadas por el alumno para modelar la situación planteada por el ejercicio son correctas. Se verifica que se indiquen todas las clases que más se ajusten a la resolución del problema así como sus atributos y métodos. El estudiante además debe identificar y establecer el morfismo a través de la justificación de la inclusión de las clases y demás componentes especificando claramente con qué parte de la letra del problema se corresponden.

A través de la asignación de diferentes puntajes a cada uno de estos elementos es posible evaluar las soluciones construidas por el estudiante con mayor grado de detalle, comprobando si identificó los componentes más importantes y justificó su aparición en la solución en forma correcta, es decir, si se estableció el morfismo adecuadamente.

c) Registros del Estudiante: El sistema almacena la información de los trabajos en curso de cada estudiante. De esta manera se le permite al estudiante realizar trabajos incrementales, agregando información a la resolución de los problemas en diferentes sesiones de trabajo. Se registran además todas las acciones realizadas durante la utilización del sistema, llevando una bitácora individual que permite analizar la forma en que interactúa con la herramienta y los pasos que sigue para construir cada solución.

d) Tutor: Este componente permite analizar las soluciones mientras se están construyendo, de manera de poder guiar al estudiante durante la resolución de los ejercicios. Mediante un conjunto de "pistas" o "guías" configuradas por el docente, la herramienta brinda ayuda al estudiante mientras construye las soluciones. Este proceso se realiza a demanda cada vez que el estudiante lo solicita. Además es el encargado de decidir, en función de los registros del estudiante y los recursos de aprendizaje disponibles los posibles ejercicios a ofrecer.

e) Recursos de aprendizaje: El sistema contiene un almacén de ejercicios (con sus soluciones) predefinidos por los docentes desde el cual se obtienen los problemas a ser planteados a los estudiantes. Es posible definir un número variable de soluciones para cada ejercicio, contemplando así variedad de posibles soluciones válidas.

f) Entrega: Este componente se encarga de transformar la información obtenida de los recursos de enseñanza a un formato apto para que el módulo de presentación lo muestre al estudiante.

g) Presentación: a través de este módulo se realiza la propia interacción entre el estudiante y el sistema.

Se utilizó una arquitectura distribuida, colocando el componente de Presentación en el cliente y el resto de los componentes en un servidor. La comunicación entre el cliente y el servidor se realiza utilizando *web services*. La implementación de la herramienta se realizó en Java 5 utilizando como entorno de desarrollo Eclipse.

5.2 Uso de "eL TIMoN"

En esta sección se describe el uso de la herramienta. El estudiante ingresa al sistema a través de una clave. Dispone de varios ejercicios. Cuando selecciona uno, le aparece una pantalla similar a la de la figura 2. La descripción del problema se presenta de forma narrativa, mientras que el alumno modela su solución utilizando el lenguaje de modelado UML (*Unified Modelling Language*).

A medida que lo va modelando, va agregando clases, métodos y atributos. Por cada elemento que agrega, además de indicar sus características (ejemplo: incorporar un atributo denominado nombre de tipo *String* a la clase Empleado), debe establecer explícitamente el vínculo o la correspondencia entre el ejercicio y el modelo. Para indicar esa correspondencia debe seleccionar el texto correspondiente y asociarlo al elemento que agregó (a través de presionar el botón "Copiar"). Ahí se está estableciendo claramente el morfismo. En cualquier momento puede solicitar una "pista", que le brinda una indicación de qué elemento agregar en función de cuáles ya están agregados. También, puede solicitar "corregir", indicando si quiere corregir además de las clases, los atributos, los métodos y, o las justificaciones. El sistema le informa el resultado de la corrección.

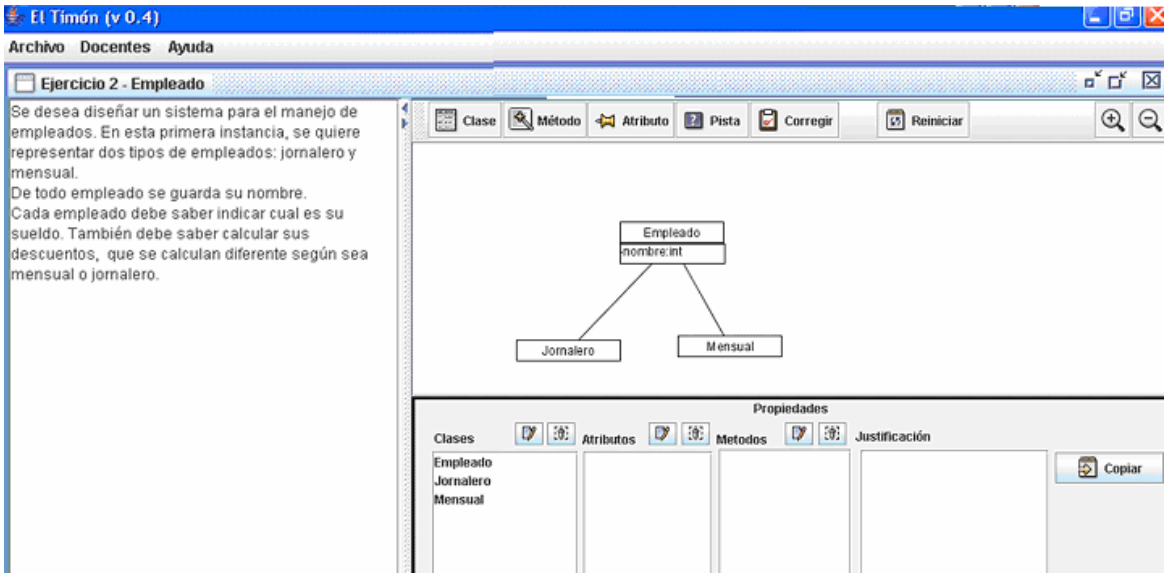

Figura 2: eL TIMoN.

El sistema guarda la solución del estudiante. Posteriormente, el estudiante puede editar la solución del ejercicio o reiniciarla completamente.

El docente dispone de una ventana en la cual ingresa letras de ejercicios y sus múltiples posibles soluciones (ver Figura 3). También puede revisar las soluciones realizadas por los estudiantes. El uso de diferentes términos para referirse a los mismos conceptos se maneja a través de "sinónimos". Así, el docente establece para cada ejercicio cuáles vocablos son semejantes o equivalentes (ejemplo: Empleado, Funcionario, Contratado).

Figura 3: eL TIMoN: área docente

6 EXPERIMENTACIÓN Y RESULTADOS

Se realizó la experimentación con 2 grupos de alumnos de Programación I. Los alumnos fueron distribuidos aleatoriamente en los grupos y también fueron asignados al azar los tratamientos. Como se presenta en la Tabla 2, al Grupo I se le brindó el curso normal, consistente en clases teóricas y prácticas así como el material teórico y de ejercicios sobre morfismos, elementos de utilidad según los resultados de la experimentación del año anterior [15]. Al Grupo II se le brindó adicionalmente el uso de la herramienta. La herramienta estuvo disponible durante todo el semestre en un sitio web específico en forma exclusiva para los alumnos del Grupo II. Durante el curso se agregaron diversos ejercicios al sistema y semanalmente se les informó a los alumnos de estos agregados.

Tabla 2: Tratamientos

Grupo	Tratamiento
I (control) - 13 alumnos	Curso normal
II (Herramienta) - 21 alumnos	Curso normal más "eL TIMoN"

El uso de la herramienta supone que permitirá desarrollar mejores modelos de software. En esta experimentación la variable independiente es el uso de la herramienta y la variable dependiente es la capacidad de modelado. Esta variable es analizada, al igual que en la experimentación 2005 referida [15], desde tres puntos de vista: a) análisis de un modelo: a partir de una realidad y de un modelo, identificar posibles problemas; b) representación de datos para casos particulares; y c) creación de un modelo de un cierto dominio.

Se realizaron dos pruebas a cada alumno de cada grupo. Cada prueba consta de 3 preguntas. El primer

test se realizó en la semana 8 del curso y el segundo al final, esto es en la semana 15. Cada pregunta del test refiere a uno de los puntos de vista referidos. Los datos que se presentan son de los alumnos que completaron ambas pruebas.

En el primer test, la primera pregunta solicita que se indique los posibles problemas de un modelo dado del mundo real en el mundo software. En particular se plantea que en el mundo real hay una fila de personas en una caja de supermercado y se modeló con una lista de objetos Persona con las operaciones: *agregar Al Principio*, *agregar Al Final*, *agregar En Posición Dada*, *sacar Del Principio* y *dar Largo De La Fila*. La segunda pregunta indica representar datos (edad de una persona, número de pasaporte, sueldo) indicando el tipo de dato y los posibles problemas que pueden darse. La tercera pregunta refiere a que se desea realizar un sistema para la gestión de un club deportivo. En el club, diversos profesores dictan clases. Los alumnos se anotan para las clases que desean tomar. Se solicita desarrollar un diagrama de clases que modele esta situación e incluir las clases que se consideren necesarias con sus atributos, métodos y relaciones. El segundo test fue diseñado en forma de evaluar el mismo contenido.

Los test se evaluaron con escalas ordinales. Cada pregunta se evaluó de 0 ("sin responder", "completamente mal") a 6 ("excelente"). Las muestras se compararon con el test del Signo y el test de Mann-Whitney. Para evaluar la primera pregunta se tuvo en cuenta la cantidad de factores o elementos indicados (por ejemplo: falta la operación abandonar la fila, falta la operación intercambiar lugares, falta modelar dejar el lugar a alguien, no están modelados los acompañantes). Para la segunda se consideró indicar el tipo de dato y la justificación. Para la tercera se tuvo en cuenta las clases indicadas, los atributos, los métodos y las diversas relaciones entre clases.

Los resultados experimentales se presentan en la Tabla 3. En relación a la pregunta 1 (análisis de un modelo) se detectó que inicialmente los grupos eran diferentes (Mann Whitney; $\alpha = 0.05, 0.10$). Analizando los datos, se constata que el grupo I tiene mejores resultados en la Prueba 1. En el Grupo II, 18 de los 23 alumnos tienen resultado 0 ó 1 (78%) mientras que en el Grupo I, solamente 4 de 13 tienen resultado 0 ó 1 (aproximadamente 30%). En la segunda prueba, con el test de Mann Whitney no se detectan diferencias significativas. De los datos individuales se constata que en el Grupo II 20 de 23 alumnos mejoran su desempeño, valor estadísticamente significativo según la prueba del signo.

En la pregunta 2, no se detectan diferencias significativas iniciales (Mann Whitney; $\alpha = 0.05, 0.10$) entre los grupos, pero sí en la segunda prueba. Los valores particulares permiten determinar que el Grupo II tiene resultados mejores: 14 de los 23 alumnos (61%) obtienen 4 o mayor puntaje, mientras que en el grupo I , 2 de 13 (15%) obtienen esta calificación.

En la pregunta 3, no se perciben diferencias significativas (Mann Whitney; $\alpha = 0.05, 0.10$) entre los grupos en ninguna de las dos pruebas. El test del signo muestra que en el Grupo II, 12 de 23 alumnos (52%) mejoran su rendimiento, valor significativo estadísticamente. En el Grupo I, 4 de 13 (31%) mejoran su desempeño.

Tabla 3 Resultados

<table>
<tr><td></td><td>Prueba inicial</td><td>Prueba Final</td></tr>
<tr><td></td><td>(Mann Whitney;
α = 0.05, 0.10).</td><td>(Mann Whitney;
α = 0.05, 0.10).</td></tr>
<tr><td>Pregunta 1

Análisis de un modelo</td><td>Diferentes

Grupo I con mejores resultados (30% de resultados bajos frente al 78% del Grupo 2)</td><td>Sin diferencias significativas entre los grupos.

Grupo II con mejoras individuales significativas (test del Signo): 87% de los alumnos mejoran:
<table><tr><td>Grupo</td><td>Mejoran</td><td>Igual</td><td>Empeoran</td></tr><tr><td>I</td><td>31%</td><td>38%</td><td>31%</td></tr><tr><td>II</td><td>87%</td><td>9%</td><td>4%</td></tr></table></td></tr>
<tr><td>Pregunta 2

Representación de datos</td><td>Sin diferencias</td><td>Con diferencias significativas.

Grupo II: 61% de los alumnos con resultados altos (valores 4,5 ó 6)
<table><tr><td>Grupo</td><td>Valores bajos</td><td>Valores altos</td></tr><tr><td>I</td><td>85%</td><td>15%</td></tr><tr><td>II</td><td>39%</td><td>61%</td></tr></table></td></tr>
<tr><td>Pregunta 3

Creación de un modelo</td><td>Sin diferencias</td><td>Sin diferencias.

Grupo II con mejoras individuales significativas - test del Signo (52% de los alumnos mejoran; 26% se mantiene igual y 22% empeora). En el Grupo I, 31% mejoran, 31% empeoran y 38% se mantiene igual.
<table><tr><td>Grupo</td><td>Mejoran</td><td>Igual</td><td>Empeoran</td></tr><tr><td>I</td><td>31%</td><td>38%</td><td>31%</td></tr><tr><td>II</td><td>52%</td><td>26%</td><td>22%</td></tr></table></td></tr>
</table>

A partir del análisis de los resultados de los test del Signo y de Mann-Whitney se puede inferir que el uso de la herramienta permite una mejora en los procesos de análisis de modelado de un dominio, la representación de datos y las habilidades para construir un modelo.

7 CONCLUSIONES Y TRABAJOS FUTUROS

Este primer uso de "eL TIMoN" permite inferir, a través de los resultados experimentales, que la herramienta es útil en el sentido de que favorece el desarrollo de estrategias para analizar y construir modelos de software adecuados. Los estudiantes que participaron en el curso en el que se utilizó la herramienta tuvieron mejores resultados en cuanto al análisis de una situación real y su modelado, la representación de datos y el modelo de un dominio, en relación al grupo de control.

A su vez y como continuación de este trabajo, para futuras versiones se incorporará a la herramienta

el uso de agentes que permitan brindar un apoyo de tutoría inteligente y personalizada y que den sugerencias sobre la resolución de cada uno de los ejercicios. Asímismo, a partir del análisis de las bitácoras de trabajos realizados y de las soluciones propuestas por los alumnos, se tratará de identificar problemas recurrentes que eventualmente aparezcan durante la resolución de los diversos ejercicios a los efectos de formular posteriormente estrategias o metodologías de trabajo apropiadas.

REFERENCIAS

[1] Ahern, T., Work in Progress: Effect of instructional design and pair programming on student performance in an introductory programming course, *Proceedings of the 35th ASEE/IEEE Frontiers in Education Conference FIE 2005* (Indianapolis, 2005):F3E11- F3E12, 2005

[2] Bailey, M., IronCode: Think-Twice, Code-Once Programming. *SIGCSE 2005,* 2005

[3] Bayliss, J. y Strout, S., Games as a "flavor" of CS1, *SIGCSE 06,* 2006

[4] Bentley, T. Learning beyond the classroom. *Educational Management & Administration*, 28(3):353-364, 2000.

[5] Bergin, J. (último acceso: Julio 16, 2006), Acerca de Juego de rol para presentar los conceptos de Orientación a Objetos. http://csis.pace.edu/~bergin/Java/RolePlay.html

[6] Börstler, J y Schulte, C. Teaching Object Oriented Modelling with CRC Cards and Roleplaying Games, *Proceedings WCCE 2005*, Cape Town, South Africa, 2005

[7] Bruce, K. Controversy on how to teach CS1: a discussion on the SIGCSE-members Mailing List, *Inroads- The SIGCSE Bulletin*, 36(4), pp. 29-35, 2004.

[8] Carlisle, M., Wilson, T., Humphries, J., y Hadfield, S. RAPTOR: a visual programming environment for teaching algorithmic problem solving. *SIGCSE 05,* 2005

[9] Cooper, S., Dann, W. y Pausch, R. Teaching Objects-first In Introductory Computer Science, *SIGCSE 2003*, 2003

[10] Esteves, M. y Mendes, A. A simulation tool to help learning of object oriented programming basics, *Proceedings of the 34th ASEE/IEEE Frontiers in Education Conference FIE 2004 (Savannah, 2004)*:F4C7-F4C12, 2004.

[11] Huet, I., Rocha, O., Tavares, J. y Weir, G., New challenges in teaching introductory programming courses: a case study, *Proceedings of the 34th. ASEE/IEEE Frontiers in Education Conference FIE 2004 (Savannah, 2004)*:T2H5-T2H9, 2004.

[12] IEEE Standard for learning technology - Learning technology systems architecture: LTSA. IEEE-1484.1-2003.

[13] Jackson, P. *Práctica de la Enseñanza.* Ed. Amorrortu, Buenos Aires, 2002.

[14] Kalling, L. y Nordström, M., Teaching OO Concepts - A New Approach, *Proceedings of the 34th. ASEE/IEEE Frontiers in Education Conference FIE 2004 (Savannah, 2004)*: F3C6-F3C11, 2004.

[15] Kereki, I, Crespo, J y Azpiazu, J. Use of Morphism as a tool to help learning object oriented concepts. A ser presentado en: *IFIP Conference on Education for the 21st Century - Impact of ICT and Digital Resources*, Chile, 2006

[16] McCombs, B. y Whisler, J. *La clase y la escuela centradas en el aprendiz. Estrategias para aumentar la motivación y el rendimiento*. Ediciones Paidós Ibérica SA, España, 2000.

[17] McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Ben-David, Y., Laxer, C., Thomas, L., Utting, I. y Wilusz, T., A multi-national, multi-institutional study of assessment of programming skills of first-year CS students, *ACM SIGCSE Bulletin*, 33(4):125-140, 2001.

[18] Pazos Sierra, J., Enseñanza del futuro: a grandes males pequeños remedios. *Universidad Politécnica de Madrid*, España, 2001.

[19] Pazos Sierra, J. *Morfismos*. Pendiente de publicación, 2004.

[20] Robins, A, Rountree, J. y Rountree, N. Learning and teaching programming: a review and discussion, *Computer Science Education* 13(2):137-172, 2003

[21] Sanders, D. and Dorn, B., Jeroo: A Tool for Teaching Object-Oriented Programming, *Proceedings of the Thirty-Fourth SIGCSE Technical Symposium*, 35 (1) 201-204, Febrero 2003

[22] Truong, N., Roe, P. y Bancroft, P. Static analysis of Student's Java programs. *Sixth Australian Computing Education Conference ACE 2004, Conferences in Research and Practice in information technology*, 30, Raymond Lister and Alison Young, Eds., New Zealand, 2004